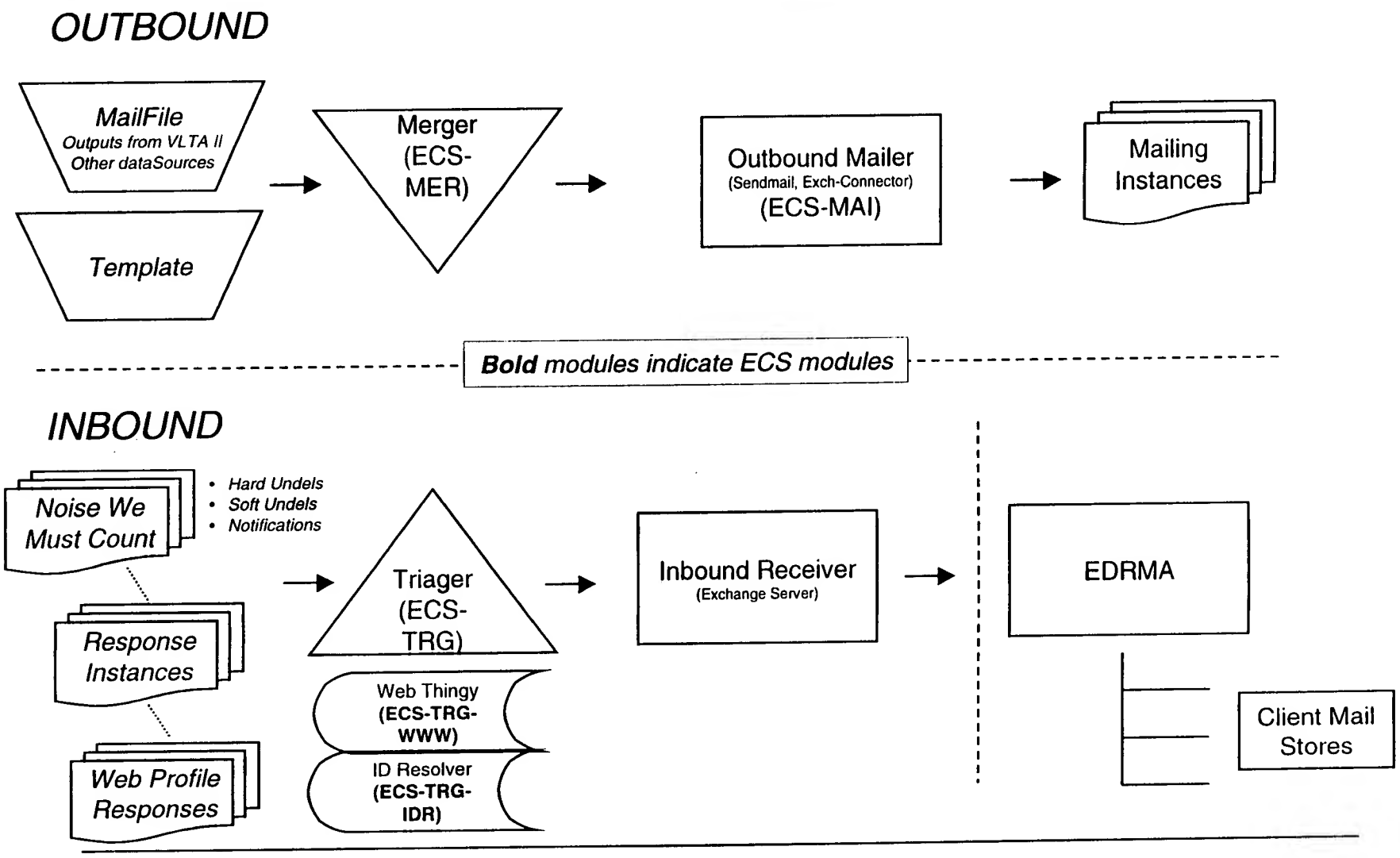
This document is intended to provide a schematic synopsis of E-Dialog Response Management Architecture (EDRMA).

Thereby it is hoped that the design of the Verbind E-Mail Channel Server (ECS) can best accommodate EDRMA's input requirements, and so EDRMA's output requirements may be adjusted to match those of Verbind LifeTime's architecture (VLTA)

"First, some acronyms…" - *Anonymous*

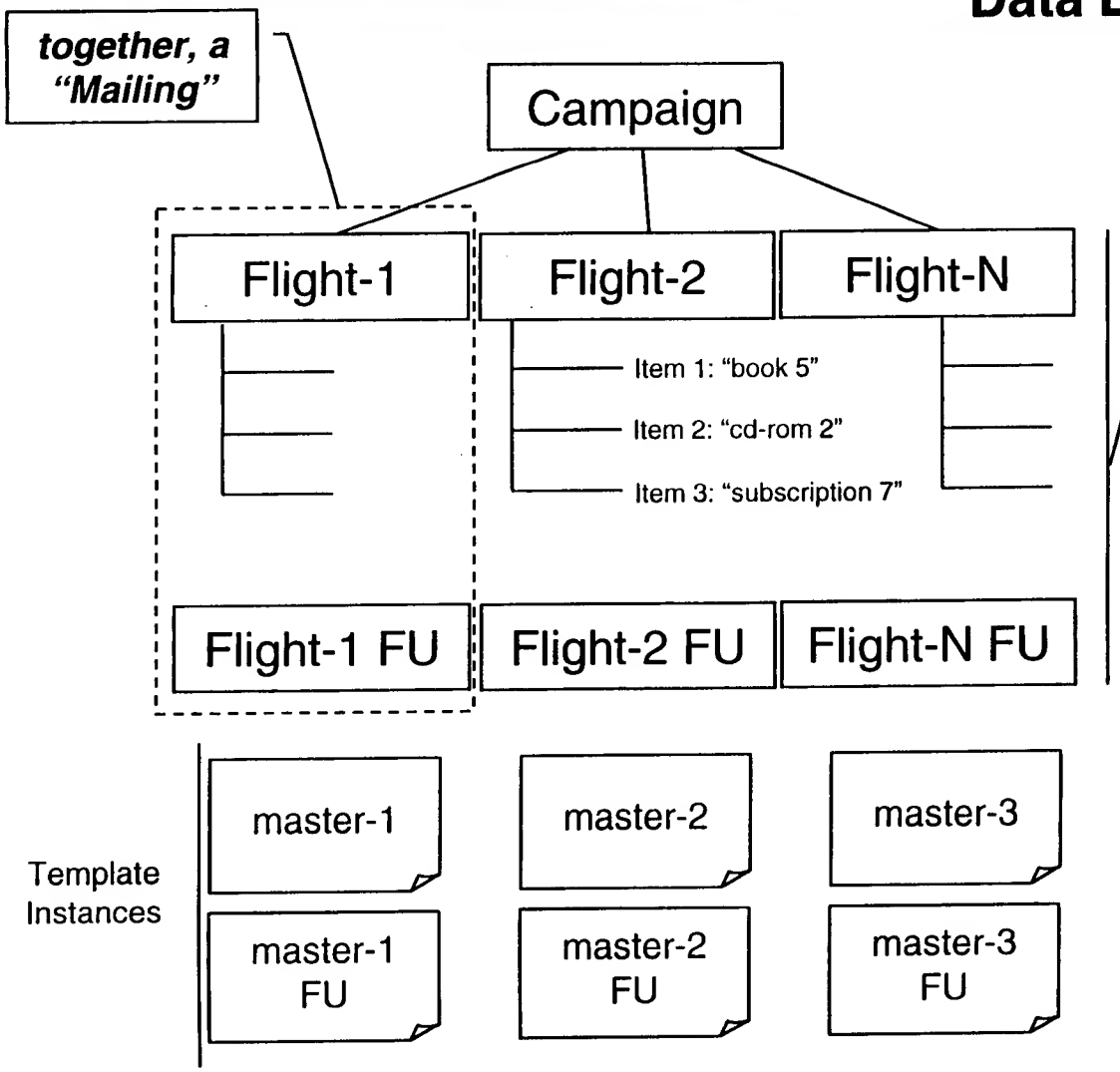
EDRMA

Relevant Modules - Top [0.1]
Where EDRMA Fits w/ECS



EDRMA Model

Data Entity Relationships [0.5]



Campaign
- has 1 CID
- has 1:N Flights (FIDs)
- has 1 template class (TCID)
  - eNews
  - QuickReply - single
  - QuickReply - multi
- Approved Response Templates (template ID)

Mail Key Codes
- [[CustID|ListID|FID|TCID] <now>
- [[**MIID**|FID|TCID]] <future>
- **MIID** -> global mail instance ID
- placement: X-Header, Subject, Footer

Mailing (e.g., F1 & F1-FU)
- has 1 MID
- have 2 FIDs, one for F, FU
- has one SOAC (client code)

Any Flight has 1:N items for sale
- has one FID
- has one TCID
- (typ) shares a SOAC with its FU
- Any Item
  - has 1 InstanceID (our code)
  - has 1 ItemID (client code)

Canonical Data Structures - Exchange Server Data Store [0.2]

Mailbox - <ClientFullName> *(not alias: "hbsp")*

<CampaignName>[<CID>]

- **ApprovedResponseTemplates <templateID>**
- **<FlightName>[<FID>]**
  - **AC** *(i.e., "Advocacy Care")*
    - + <date: YYMMDDhhmm>
  - **ADDCHANGE**
    - + <date: YYMMDDhhmm>
  - **HARDBOUNCES** *(i.e., hard undelivereds... "jsmith@foo.com is not a valid addressee")*
    - + <date: YYMMDDhhmm>
  - **INBOX (monolithic, this Mailbox)**
  - **MASTER (contains master template, this FID)**
  - **ORDERS**
    - + <date: YYMMDDhhmm>
      - » [[custIDIListIDIFlightIDITCID]] << footer tags (x-header, subject)
  - **SOFTBOUNCES**
    - + **DELIVERYNOTIFICATIONS**
      - » <date: YYMMDDhhmm>
    - + **AUTORESPONDERS**
      - » <date: YYMMDDhhmm>
    - + **UNKNOWN**
      - » <date: YYMMDDhhmm>
  - **UNCLEAR**
    - + <date: YYMMDDhhmm>
  - **UNSUBS**
    - + <date: YYMMDDhhmm>

EDRMA

Selected Applications

VBA.PKinboxInspector
interface: GUI / VBA
data: MAPI
inbox sorting, folder management application

VBA.PKresponseProcessor
interface: GUI / VBA
data: MAPI
response review and report preparation application

PL.AEprocOrder()
interface: commandline
data: ADO 2.0
process raw "order e-mails" by TCID
uses cf file rulesets for document preprocessing and data element parsing tied to CID; other rules, hardcoded
generates:
Acceptable output for review, annotated
Exceptions, annotated
Truncated BODYs, annotated
Raw fields output , annotated
Raw fields exceptions , annotated
Rule-eval log (exhaustive)

PL.AEprocBatPrep()
interface: commandline
data: ADO 2.0
takes selected output from PL.AEprocOrder() and transforms to a batch transfer specification via a field map and transform rules cf

PL.AEprocUpdateBatVerbind()
TBD

PL.AEscrubNormalCanon()
<< not used on the response side >>
interface: commandline
data: file handles
scrubber, normalizer, canonicalizer
also generates scrambled (non-predictable) row Ids and flags AOLs

Process Stages [0.5]

Preliminary ECS hooks anticipated

Preprocess Stage

- inspect Inbox using VBA.PKinboxInspector
  - auto-creates folder structure (previous slide)
  - facilitates auto-sort of items into ORDERS, UNSUBS, UNDELS, HARDS, SOFTS, ADDCHANGE, etc.
  - facilitates manu-sort of items into AC, and exception
- report preparation by TCID via AEprocOrder()
  - produces: OUT_report, EXC_report && EXC_BODY diagnostic report

Processing Stage using VBA.PKresponseProcessor

- for each EXC_
  - inspect, correct, reconcile (via Mailing Table lookups) and commit
- for each OUT_
  - inspect and commit to report (or not)
- Acceptable UNION of EXC_ && OUT_ -> Reporting, Update Stages { AEprocBatPrep() || AEprocUpdateBatVerbind() }
- Hard exceptions are tagged as such and become Followup RFC's to get additional (critical information)
- FUP (FUPID -> RFC) tags affect routing; tie-back to HARD exception row in this EXC_ report for closure

Response Follow-up Confirmation Stage

- for each non-BOUNCE (ORDERS, UNSUBS, etc.), respond to respondents with an appropriate "confirmation of receipt/action taken" message
- using relevant response template store **(templateID)**
- this class of response likewise tagged for routing (FUPID -> confirm)

Reporting Stage

- produce order report
  - deliver via fax
  - post to client private web application
  - deliver as formatted batch

Update Stage

- deliver formatted batch to {Verbind, Database} -> sp_???, SQL Executive